



## **Secure Firewall eStreamer Fully-Qualified Events Guide for 7.4**

**First Published:** 2023-10-19

### **Americas Headquarters**

Cisco Systems, Inc.  
170 West Tasman Drive  
San Jose, CA 95134-1706  
USA  
<http://www.cisco.com>  
Tel: 408 526-4000  
800 553-NETS (6387)  
Fax: 408 527-0883





## CONTENTS

---

### CHAPTER 1

#### **Introduction 1**

Using this Guide 1

Prerequisites 2

IP Addresses 2

Best Practices 2

---

### CHAPTER 2

#### **Understanding the eStreamer Protocol 5**

Connection Specifications 5

Establishing an Authenticated Connection 6

Establishing a Session 6

Error Message Format 7

Requesting Fully-qualified Events 7

Format of the JSON file 8

Fully-Qualified Event Messages 9

Accepting Data from eStreamer 10

Changing a Request 10

Terminating Connections 10

---

### CHAPTER 3

#### **Available Fully Qualified Events 11**

Connection Event 11

File Event 28

Intrusion Event 37

Intrusion Packets 51





# CHAPTER 1

## Introduction

---

The Cisco Event Streamer (also known as eStreamer) allows you to stream Firepower System events to external client applications. While eStreamer continues to support the binary streaming of events, you may also request fully-qualified events. These events are in a clear text format with metadata resolved. This guide describes how to request these fully-qualified events from the eStreamer service.

Connection events, intrusion events, intrusion packets, and file events are available as fully-qualified events from a Management Center.

Note that eStreamer is not supported on NGIPSv, Firepower Services, Firepower Threat Defense Virtual, and Firepower Threat Defense. To stream events from these devices, you can configure eStreamer on the Management Center that the device reports to.

eStreamer uses a custom application layer protocol to communicate with connected client applications. As the purpose of eStreamer is simply to return data that the client requests, the majority of this guide describes the eStreamer formats for the requested data.

There are four major steps to creating and integrating an eStreamer client with a Firepower System:

1. Write a client application that exchanges messages with the Management Center or managed device using the eStreamer application protocol.
2. Configure a Management Center or device to send the required type of events to your client application.
3. Connect your client application to the Management Center or device.
4. Specify the data and format you want and begin exchanging data.

This guide provides the information you need to successfully create and run an eStreamer client application which receives fully-qualified events.

- [Using this Guide, on page 1](#)
- [Prerequisites, on page 2](#)
- [IP Addresses, on page 2](#)
- [Best Practices, on page 2](#)

## Using this Guide

- At the highest level, the eStreamer service is a mechanism for streaming data from the Secure Firewall System to a requesting client. The service can stream the following fully-qualified events:
  - Connection Events

- Intrusion Events
- Intrusion Packets
- File Events

Descriptions of the data structures returned by eStreamer make up the majority of this book. The chapters in the book are:

## Prerequisites

To understand the information in this guide, you should be familiar with the features and nomenclature of the Secure Firewall System and the function of its components in general, and with the different types of event data these components generate in particular. Definitions of unfamiliar or product-specific terms can frequently be obtained from the *Secure Firewall eStreamer Integration Guide*.

You should also be familiar with CSV (comma-separated values) or JSON (JavaScript Object Notation) file format and be able to create a program or script which can use one of these formats.

## IP Addresses

The Cisco database stores IPv4 and IPv6 addresses in the same fields in a hexadecimal format. To get IPv6 addresses, convert to hex notation, for example: 20010db800000000000000000000004321. The database follows the RFC for storing IPv4 addresses by filling in bits 80-95 with 1's, which yields an invalid IPv6 address. For example, the IPv4 address 10.5.15.1 would be stored as 000000000000000000000000FFFF0A050F01.

## Best Practices

When working with eStreamer, Cisco recommends the following for best use of the API.

### Design

- Build your eStreamer client to support everything the API can provide, as every bit of the schema is important to at least some small part of the customer base.
- Implement robust error handling and logging so that when something goes wrong, you can see the message and situation that caused the problem without necessarily needing to reproduce the error.
- Pick your language carefully. Parsing might not seem that computationally expensive but when there are thousands of events per second, everything counts. Compiled languages such as C, C++, Go will be faster than Python / JavaScript. The downside of such an approach is lack of portability.
- Look at the existing eStreamer implementations to see how others have accomplished your goals in the past. Some resources:

<https://splunkbase.splunk.com> and search for eStreamer

<https://software.cisco.com/download/home/> next to “Select a Product” select “Browse All”, then “Security”, followed by “Firewalls”, “Firewall Management”, “Firepower Management Center Virtual Appliance”, then “Firepower System Tools and APIs”.

<https://community.cisco.com> and search for “eNcoreCLI”.

- Make sure to work with the Cisco Security Technical Alliance team to keep up with changes to eStreamer and other aspects of integrating with Cisco Firepower. You can contact them at [ask-csta-pm@cisco.com](mailto:ask-csta-pm@cisco.com)

### Testing

- When Cisco introduces a new version of Firepower, promptly test your client against it to make sure the data collected by your client does not change.
- Have a good test bed so you can test easily and frequently.
- If you prefer to not build your own test bed, use the dcloud sandbox test bed. The Cisco Security Technical Alliance will provide resources to assist in setting up and using it. Dcloud is free and enables comprehensive testing. However, it is not necessarily complete for your use and does not have 100% event coverage. Also, instances are only available for short periods of time. For more information about dcloud, go to <https://dcloud2-rtp.cisco.com>







## CHAPTER 2

# Understanding the eStreamer Protocol

---

The Secure Firewall System Event Streamer (eStreamer) uses a message-oriented protocol to stream events and host profile information to your client application. Your client can request fully-qualified events from a Management Center. Only connection events, intrusion events, intrusion event packets, and file events are available as fully-qualified events.

Your client application initiates the data stream by submitting request messages, which specify the data to be sent, and then controls the message flow from the Management Center or managed device after streaming begins.

Throughout this document, the eStreamer service on the Management Center or a managed device may be referred to as the eStreamer server or eStreamer.

- [Connection Specifications, on page 5](#)
- [Establishing an Authenticated Connection, on page 6](#)
- [Establishing a Session, on page 6](#)
- [Error Message Format, on page 7](#)
- [Requesting Fully-qualified Events, on page 7](#)
- [Format of the JSON file, on page 8](#)
- [Fully-Qualified Event Messages, on page 9](#)
- [Accepting Data from eStreamer, on page 10](#)
- [Changing a Request, on page 10](#)
- [Terminating Connections, on page 10](#)

## Connection Specifications

- Communicates using TCP over an SSL connection (the client application must support SSL-based authentication).
- Accepts connection requests on port 8302.
- Waits for the client to initiate all communication sessions.
- Writes all message fields in network byte order (big endian).

## Establishing an Authenticated Connection

Before a client can request data from eStreamer, the client must initiate an SSL-enabled TCP connection with the eStreamer service. The client can request on any configured management interface on the Management Center or managed device. Client connections do not enforce traffic channel configuration for management interfaces so that configuration can be ignored when choosing an interface for your connection. When the client initiates the connection, the eStreamer server responds, initiating an SSL handshake with the client. As part of the SSL handshake, the eStreamer server requests the client's authentication certificate, and verifies that the certificate is valid (signed by the Internal Certifying Authority [Internal CA] on the eStreamer server).

Cisco recommends that you also require your client to verify that the certificate presented by the eStreamer server has been signed by a trusted Certifying Authority. This is the Internal CA certificate included in the PKCS#12 file that Cisco provides when you register a new eStreamer client with the Management Center or managed device.

After the SSL session is established, the eStreamer server performs an additional post-connection verification of the certificate. This includes verifying that the client connection originates from the host specified in the certificate and that the subject name of the certificate contains the appropriate value. If either post-connection check fails, the eStreamer server closes the connection. If necessary, you can configure the eStreamer service so that it does not perform a client host name check.

While the client is not required to perform post-connection verification, Cisco recommends that the client perform this verification step. The authentication certificate contains the following field values in the subject name of the certificate:

**Table 1: Certificate Subject Name Fields**

Field	Value
title	eStreamer
generationQualifier	server

After the post-connection verification is finished, the eStreamer server awaits a data request from the client.

## Establishing a Session

The client establishes a session by sending an initial Event Stream request to the eStreamer service.

For fully-qualified events you must submit the data requests in a follow-on message. This initial Event Stream request message itself is a prerequisite for all eStreamer requests.




---

**Note** The eStreamer client can request on any configured management interface on the Management Center or managed device. Client connections do not enforce traffic channel configuration for management interfaces so that configuration can be ignored when choosing an interface for your connection.

---

## Error Message Format

Both the client application and the eStreamer service use error messages. Error messages have a message type of 1 and contain a header, an error code, an error text length, and the actual error text. Error text can contain between 0 and 65,535 bytes.

When you create custom error messages for your client application, Cisco recommends using -1 as the error code.

The following table describes the basic error message format. The Header Version, Message Type, and Message Length are not specific to the error message type.

Bytes	Name	Data Type	Description
0-15	Header Version	uint16	Always 1.
16-31	Message Type	uint16	Always 1.
32-63	Message Length	uint32	Length of the error message, in bytes.
64-95	Error Code	uint32	A number representing the error.
96-111	Error Text Length	uint16	The number of bytes included in the error text field.
112 on	Error Message	Variable	The error message. Up to 65,535 bytes.

## Requesting Fully-qualified Events

Instead of receiving events in the complicated binary format, we recommend that your client uses this option to request fully-qualified events in a text format such as JSON or CSV. When using this option, the majority of this document describing the binary format is irrelevant. In the SDK package, the `python_client` subdirectory provides sample code for using this option.

This option currently only supports requesting information for a few event types: connection events, intrusion events, intrusion packets, and file events. If you need to receive other event types in binary format, then separate client connections must be used for fully-qualified and binary event formats.

To request fully-qualified events, use the documented "Event Stream Request Message", and append a JSON-format configuration block at the end of the message. The request will include the usual five binary integers shown below, followed by the JSON-format configuration details, like:

```
<Header Version (1)> <Message Type (2)> <Message Length> <Initial Timestamp> <Request Flags>
<JSON-format Configuration Block>
```

The binary Message Length field must include the length of the binary header, plus the length of the JSON block. A terminating null character is optional after the JSON block, but if the null is included then the Message Length must account for the null character. For the Request Flags field, only bit 23 (extended event headers) is supported; all other bits should be zero, in particular bit 30 (extended request) must be zero.

After the client sends the request message, the eStreamer service will immediately start sending event data if the requested event types have been enabled on the server side UI eStreamer configuration page.

## Format of the JSON file

This example can be also found in the `json_request.json` file in the eStreamer SDK.

```
{ "Events":
```

This section specifies the requested fields from connection events. If this section were removed, the eStreamer server would not send any connection events.

```
{ "ConnectionEvent":
  { "FieldSetDef":
    { "OutputFieldSet":
      ["HeaderFieldSet", "ConnectionKeySet", "DetailFieldSet"] },
    "Fields":
      ["OutputFieldSet"] },
```

This section specifies the requested fields from intrusion events. If this section were removed, the eStreamer server would not send any intrusion events.

```
"IntrusionEvent":
  { "FieldSetDef":
    { "OutputFieldSet":
      ["HeaderFieldSet", "ConnectionKeySet", "DetailFieldSet", "Impact"] },
    "Fields": ["OutputFieldSet"] },
```

This section specifies the requested fields from intrusion event packets. If this section were removed, the eStreamer server would not send any intrusion event packets.

```
"IntrusionPacket":
  { "FieldSetDef":
    { "OutputFieldSet":
      ["HeaderFieldSet", "DetailFieldSet"] },
    "Fields": ["OutputFieldSet"] },
```

This section specifies the requested fields from file events. If this section were removed, the eStreamer server would not send any file events.

```
"FileEvent":
  { "FieldSetDef":
    { "OutputFieldSet":
      ["HeaderFieldSet", "ConnectionKeySet", "DetailFieldSet"] },
    "Fields":
      ["OutputFieldSet"] } },
```

This section specifies the output format as described below.

```
"OutputFormat":
  { "Transform": "Text", "TransformConfig": "JSON" } }
```

In the `Events` section, specify a block for each event type that you would like the client to receive (only the three example types are supported: `ConnectionEvent`, `IntrusionEvent`, `IntrusionPacket`, and `FileEvent`). The `FieldSetDef` section for each event must specify an `OutputFieldSet`, which lists the fields or field sets which will be included in the events for that event type. The sample file only specifies field sets, but you can use any combination of field names and field sets.

The list of available fields for each event type, and the predefined field sets, can be found on the Firepower Management Center in the file `/etc/sf/EventHandler/EventCatalog/EventCatalog.json`. In the Fields section towards the end of the file, look for the desired event type (such as `IntrusionEvent`), then see the `Fields` and `FieldSetDef` blocks to see what is available for that event type.

The `OutputFormat` section has settings for the output. The `Transform` field is always `Text`, and you specify the output transformation format with the `TransformConfig` field. The example shows `JSON`, but you can also specify `CSV`. Other text formats are available, as well as `FlatBuffer`, but you will need to request documentation for these formats.

When `JSON` output is specified in `TransformConfig`, the output will contain name-value pairs for each requested field, except any fields which are irrelevant to the event are skipped (e.g. if you requested `SSL` fields, and an event did not use `SSL`, then the output will not contain those fields).

When `CSV` output is specified in `TransformConfig`, the output will contain the desired fields in the order listed in the configuration. If a field is not relevant to the event then the `CSV` will only contain a comma for that field. Do not use predefined field sets when requesting `CSV` because the field sets may change between versions, making the `CSV` incompatible.

## Fully-Qualified Event Messages

Event messages are contained in bundles, as described in the eStreamer documentation for "Message Bundle Format", message type 4002.

The client must acknowledge each received data bundle by sending a null message to the eStreamer server, indicating readiness to accept more data.

For all supported event types, the event data message starts with the binary header that is described in the eStreamer documentation for various event types, such as the "Intrusion Record Header". The only difference is that the data block format is the requested format (`JSON`, `CSV`, etc.). For quick reference the basic structure is:

```
<Header Version (1)>
<Message Type (3)>
<Message Length>
<Record Type (with optional Netmap ID when requested)>
<Record Length> <Timestamp (when request bit 23 is specified)>
<Reserved (when request bit 23 is specified)>
<Data>
```

# Accepting Data from eStreamer



---

**Note** The eStreamer server does not keep a history of the events it sends. Your client application must check for duplicate events, which can inadvertently occur for a number of reasons. For example, when starting up a new streaming session, the time specified by the client as the starting point for the new session can have multiple messages, some of which may have been sent in the previous session and some of which were not. eStreamer sends all message that meet the specified request criteria. Your application should detect any resulting duplicates.

---

During periods of inactivity, eStreamer sends periodic null messages to the client to keep the connection open. If it receives an error message from the client or an intermediate host, it closes the connection.

eStreamer transmits requested data to the client differently, depending on the request mode.

## Changing a Request

To change request parameters for an established session, the client must disconnect and request a new session.

## Terminating Connections

The eStreamer server attempts to send an error message before closing the connection. For information on error messages, see [Error Message Format](#).

The eStreamer server can close a client connection for the following reasons:

- Any time sending a message results in an error. This includes both event data messages and the null keep-alive message eStreamer sends during periods of inactivity.
- An error occurs while processing a client request.
- Client authentication fails (no error message is sent).
- eStreamer service is shutting down (no error message is sent).

Your client can close the connection to eStreamer server at any time and should attempt to use the error message format to notify the eStreamer server of the reason.



## CHAPTER 3

# Available Fully Qualified Events

This chapter provides details about the fully-qualified events which can be requested through the Event Streamer API.

- [Connection Event, on page 11](#)
- [File Event, on page 28](#)
- [Intrusion Event, on page 37](#)
- [Intrusion Packets, on page 51](#)

## Connection Event

Connection Events are generated when the system detects a connection. These events record a variety of information about the connection depending on the type of connection.

You can request fully-qualified connection events from the eStreamer service by including a "ConnectionEvent" section in the request JSON file. This section must specify the requested fields. The following fields may be requested for connection events:

- `[FirewallRuleList]`  
List which contains the firewall rule which triggered the event and all other matching monitor rules.
- `[MonitorRule]`  
List of all Access Control rules which match the event.
- `[MonitorRuleID]`  
List of the ID numbers for the monitor rules which match the event.
- `AC_RuleAction`  
The action associated with the configuration that logged the connection.

For Security Intelligence-monitored connections, the action is that of the first non-Monitor access control rule triggered by the connection, or the default action. Similarly, because traffic matching a Monitor rule is always handled by a subsequent rule or by the default action, the action associated with a connection logged due to a Monitor rule is never Monitor. However, you can still trigger correlation policy violations on connections that match Monitor rules.

Action	Description
Allow	Connections either allowed by access control explicitly, or allowed because a user bypassed an interactive block.
Block, Block with reset	Blocked connections, including: <ul style="list-style-type: none"> <li>• tunnels and other connections blocked by the prefilter policy</li> <li>• connections blocked by Security Intelligence.</li> <li>• encrypted connections blocked by an SSL policy.</li> <li>• connections where an exploit was blocked by an intrusion policy.</li> <li>• connections where a file (including malware) was blocked by a file policy.</li> </ul> For connections where the system blocks an intrusion or file, system displays <code>Block</code> , even though you use access control <code>Allow</code> rules to invoke deep inspection.
Fastpath	Non-encrypted tunnels and other connections fastpathed by the prefilter policy.
Interactive Block, Interactive Block with reset	Connections logged when the system initially blocks a user's HTTP request using an Interactive Block rule. If the user clicks through the warning page that the system displays, additional connections logged for the session have an action of <code>Allow</code> .
Trust	Connections trusted by access control. The system logs trusted TCP connections differently depending on the device model.
Default Action	Connections handled by the access control policy's default action.
(Blank/empty)	The connection closed before enough packets had passed to match a rule.  This can happen only if a facility other than access control, such as intrusion prevention, causes the connection to be logged.

- `AC_RuleActionID`

ID number of the access control rule action.

- `AC_RuleReason`

The reason or reasons the connection was logged, in many situations. For a full list, see [Connection Event Reasons](#).

Connections with a Reason of IP Block, DNS Block, and URL Block have a threshold of 15 seconds per unique initiator-responder pair. After the system blocks one of those connections, it does not generate connection events for additional blocked connections between those two hosts for the next 15 seconds, regardless of port or protocol.

- `AC_RuleReasonID`

ID number of the Access Control Rule reason.

- `Application`



The application protocol, if available, which represents communications between hosts detected in the traffic that triggered the intrusion event.

- `ApplicationID`

ID number of the application.

- `ApplicationProductivityIndex`

• Business relevance of the application. Values may be:

- 1 - Very Low
- 2 - Low
- 3 - Medium
- 4 - High
- 5 - Very High

- `ApplicationProtocolNegotiations`

TLS Value used to negotiate protocols for data transfer. This value is defined in RFC 7301.

- `ApplicationRiskIndex`

The risk associated with detected applications in the traffic that triggered the intrusion event: Very High, High, Medium, Low, and Very Low. Each type of application detected in a connection has an associated risk; this field displays the highest risk of those.

- `AuthenticationSource`

Type of authentication used by the user. Values may be:

- 0 - no authorization required
- 1 - passive authentication, AD agent, or ISE session
- 2 - captive portal successful authentication
- 3 - captive portal guest authentication
- 4 - captive portal failed authentication

- `ClientAppDetector`

The sensor which detected the client.

- `ClientAppDetectorID`

ID of the sensor which detected the client.

- `ClientApplication`

Name of the client application.

- `ClientApplicationID`

The internal identification number for the client application, if applicable.

- `ClientApplicationProductivityIndex`

Business relevance of the application. Values may be:

- 1 - Very Low
- 2 - Low
- 3 - Medium
- 4 - High
- 5 - Very High

- `ClientApplicationRiskIndex`

Risk value of the client application. Values may be:

- 1 - Very Low
- 2 - Low
- 3 - Medium
- 4 - High
- 5 - Very High

- `ClientApplicationVersion`

The client application and version of that client detected in the connection.

If the system cannot identify the specific client used in the connection, the field displays the word "client" appended to the application protocol name to provide a generic name, for example, FTP client.

- `ConnectionDuration`

This field exists ONLY as a syslog field; it does not exist in the Firepower Management Center web interface. (The web interface conveys this information using the First Packet and Last Packet columns.)

This field has a value only when logging occurs at the end of the connection. For a start-of-connection syslog message, this field is not output, as it is not known at that time.

For an end-of-connection syslog message, this field indicates the number of seconds between the first packet and the last packet, which may be zero for a short connection. For example, if the timestamp of the syslog is 12:34:56 and the ConnectionDuration is 5, then the first packet was seen at 12:34:51.

- `ConnectionID`

A unique identifier for the connection with the server.

- `Context`

The metadata identifying the virtual firewall group through which the traffic passed. The system only populates this field for ASA FirePOWER in multiple context mode.

- `DestinationIP_DynamicAttribute`

Dynamic Attributes associated with the destination IP address.

- `DestinationSecurityGroup`

This field holds the text value associated with the numeric value in **DestinationSecurityGroupTag**, if available. If the group name is not available as a text value, then this field contains the same integer value as the DestinationSecurityGroupTag field.

- `Device`

In the Firepower Management Center web interface, this value constrains summaries and graphs.

The managed device that detected the connection or, for connections generated from NetFlow data, the managed device that processed the data.

- `DeviceIP`

IP address of the device that detected the event.

- `DeviceSerialNumber`

Serial number of the device that detected the event.

- `DeviceUUID`

The unique identifier of the Firepower device that generated an event.

The following fields collectively uniquely identify a connection event: DeviceUUID, First Packet Time, Connection Instance ID, and Connection Counter.

- `DNS_Query`

The DNS query submitted in a connection to the name server to look up a domain name.

This field can also hold the domain name for URL filtering matches when DNS filtering is enabled. In this case, the URL field will be blank and the URL Category and URL Reputation fields contain the values associated with the domain.

For more information about DNS filtering, see [DNS Filtering: Identify URL Reputation and Category During DNS Lookup](#).

- `DNS_RecordDescription`

Description of the DNS Record.

- `DNS_RecordType`

The type of the DNS resource record used to resolve a DNS query submitted in a connection.

- `DNS_RecordTypeID`

ID number associated with the DNS Record Type.

- `DNS_ResponseType`

The type of the DNS resource record used to resolve a DNS query submitted in a connection.

- `DNS_ResponseTypeID`

ID number associated with the DNS Response Type.

- `DNS_Sinkhole`

The name of the sinkhole server where the system redirected a connection.

- `DNS_SinkholeUUID`

The UUID of the sinkhole server where the system redirected a connection.

- `DNS_TTL`

The number of seconds a DNS server caches the DNS resource record.
- `Domain`

The domain of the managed device that detected the connection or, for connections generated from NetFlow data, the domain of the managed device that processed the data. This field is only present if you have ever configured the FMC for multitenancy.
- `DynamicAttributes`

List of dynamic attributes available for security policies.
- `EgressInterface`

The ingress or egress interface associated with the connection. If your deployment includes an asymmetric routing configuration, the ingress and egress interface may not belong to the same inline pair.
- `EgressInterfaceUUID`

An interface ID that acts as the unique identifier for the egress interface associated with correlation event.
- `EgressVRF`

In networks using virtual routing, the names of the virtual routers through which traffic entered and exited the network.
- `EgressZone`

The ingress or egress security zone associated with the connection.
- `EgressZoneUUID`

A zone ID that acts as the unique identifier for the egress security zone associated with correlation event.
- `EndpointProfile`

The user's endpoint device type, as identified by ISE.
- `EndpointProfileID`

ID number of the type of device used by the connection endpoint as identified by ISE. This is unique for each DC and resolved in metadata.
- `EVE_Fingerprint` `EVE_Process`

The TLS fingerprint detected by the Encrypted Visibility Engine (EVE) for the session.
- `EVE_ProcessConfidencePct`

The confidence value in the range 0-100% that the encrypted visibility engine has detected the right process. For example, if the process name is Firefox and if the confidence score is 80%, it means that the engine is 80% confident that the process it has detected is Firefox.
- `EVE_ThreatConfidenceIndex`

The probability level that the process detected by the encrypted visibility engine contains threat. This field indicates the bands (Very High, High, Medium, Low, or Very Low) based on the value in the threat confidence score.
- `EVE_ThreatConfidencePct`

The confidence value in the range 0-100% that the process detected by the encrypted visibility engine contains threat. If the threat confidence score is very high, say 90%, then the Encrypted Visibility Process Name field displays "Malware."

- `EventPriority`

Whether or not the connection event is a high priority event. `High` priority events are connection events that are associated with an intrusion, Security Intelligence, file, or malware event. All other events are `Low` priority.

- `EventSecond`

UNIX timestamp (seconds since 01/01/1970) of the event's detection.

- `EventSubtype`

The sub-type of malware event.

- `FileCount`

The number of files (including malware files) detected or blocked in a connection associated with one or more file events.

- `FirewallPolicy`

Name of the access control policy.

- `FirewallPolicyUUID`

UUID of the access control policy.

- `FirewallRule`

Access control rule that created the connection event.

- `FirewallRuleID`

A rule ID number that acts as a unique identifier for the access control rule.

- `FirstPacketSecond`

UNIX timestamp of the date and time the first packet was exchanged in the session.

- `Hostname`

Domain name of the detected HTTP request.

- `HTTP_Referer`

The HTTP referrer, which represents the referrer of a requested URL for HTTP traffic detected in the connection (such as a website that provided a link to, or imported a link from, another URL).

- `HTTP_Response`

The HTTP status code sent in response to a client's HTTP request over a connection.

- `ICMP_Code`

In the Firepower Management Center web interface, these values constrain summaries and graphs.

The port or ICMP code used by the session responder.

- `ICMP_Type`

In the Firepower Management Center web interface, these values constrain summaries and graphs.

The port or ICMP type used by the session initiator.

- `IngressInterface`

The ingress or egress interface associated with the connection. If your deployment includes an asymmetric routing configuration, the ingress and egress interface may not belong to the same inline pair.

- `IngressInterfaceUUID`

An interface ID that acts as the unique identifier for the ingress interface associated with correlation event.

- `IngressVRF`

In networks using virtual routing, the names of the virtual routers through which traffic entered and exited the network.

- `IngressZone`

The ingress or egress security zone associated with the connection.

For rezoned encapsulated connections, the ingress field displays the tunnel zone you assigned, instead of the original ingress security zone. The egress field is blank.

- `IngressZoneUUID`

A zone ID that acts as the unique identifier for the ingress security zone associated with correlation event.

- `InitiatorBytes`

The total number of bytes transmitted by the session initiator or received by the session responder.

- `InitiatorBytesDropped`

The number of bytes dropped from the session initiator or session responder due to rate limiting.

- `InitiatorContinent`

When a routable IP is detected, the continent associated with the IP address for the session initiator or responder.

- `InitiatorContinentCode`

ISO-3166 code for the continent of the source host.

- `InitiatorCountry`

When a routable IP is detected, the country associated with the IP address of the session initiator or responder. The system displays an icon of the country's flag, and the country's ISO 3166-1 alpha-3 country code. Hover your pointer over the flag icon to view the country's full name.

- `InitiatorCountryCode`

Code for the country of the source host.

- `InitiatorCountryID`

ISO-3166 value for the country of the initiating host.

- `InitiatorIP`

In the Firepower Management Center web interface, these values constrain summaries and graphs.

The IP address (and host name, if DNS resolution is enabled) of the session initiator or responder.

See also [A Note About Initiator/Responder, Source/Destination, and Sender/Receiver Fields](#).

In the Firepower Management Center web interface, the host icon identifies the IP address that caused the connection to be blocked.

For plaintext, passthrough tunnels either blocked or fastpathed by the prefilter policy, initiator and responder IP addresses represent the tunnel endpoints—the routed interfaces of the network devices on either side of the tunnel.

- `InitiatorPackets`

The total number of packets transmitted by the session initiator or received by the session responder.

- `InitiatorPacketsDropped`

The number of packets dropped from the session initiator or session responder due to rate limiting.

- `InitiatorPort`

Port used by the initiating host.

- `InstanceID`

umerical ID of the Snort instance on the managed device that generated the event.

- `IntrusionCount`

Number of intrusions that have been triggered by this connection.

- `IOC_Count`

Number of indications of compromise that have been triggered by this connection.

- `ManagerName`

Name of the Secure Firewall Management Center which detected the event.

- `ManagerUUID`

UUID of the Secure Firewall Management Center which detected the event.

- `MatchedRule`

The specific rule which triggered the event.

- `NAP_Policy`

The network analysis policy (NAP), if any, associated with the generation of the event.

- `NAP_PolicyUUID`

The UUID of the Network Analysis Policy that created the intrusion event.

- `NAT_InitiatorIP`

The NAT translated IP address of the session initiator or responder.

- `NAT_InitiatorPort`

The NAT translated port of the session initiator or responder.

- `NAT_ResponderIP`

The NAT translated IP address of the session initiator or responder.

- `NAT_ResponderPort`

The NAT translated port of the session initiator or responder.

- `NetBIOS_Domain`

The NetBIOS domain used in the session.

- `NetflowDestinationAS`

For connections generated from NetFlow data, the border gateway protocol autonomous system number for the source or destination of traffic in the connection.

- `NetflowDestinationTOS`

For connections generated from NetFlow data, the setting for the type-of-service (TOS) byte when connection traffic entered or exited the NetFlow exporter.

- `NetflowSNMP_In`

For connections generated from NetFlow data, the interface index for the interface where connection traffic entered or exited the NetFlow exporter.

- `NetflowSNMP_Out`

For connections generated from NetFlow data, the interface index for the interface where connection traffic entered or exited the NetFlow exporter.

- `NetflowSourceAS`

For connections generated from NetFlow data, the border gateway protocol autonomous system number for the source or destination of traffic in the connection.

- `NetflowSourceTOS`

For connections generated from NetFlow data, the setting for the type-of-service (TOS) byte when connection traffic entered or exited the NetFlow exporter.

- `NetmapID`

The first bit of this field is a flag indicating whether the header is an extended header containing an archive timestamp. The remaining 15 bits are an optional field containing the Netmap ID for the domain on which the event was detected. If this field is not used, it is left empty. Netmap IDs map to domains as provided in metadata.

- `OriginalInitiatorIP`

Contains the IP address of the original initiator of the connection.

- `PrefilterPolicy`

The prefilter policy that handled the connection.

- `Protocol`

In the Firepower Management Center web interface:

- This value constrains summaries and graphs.
- This field is available only as a search field.



The transport protocol used in the connection. To search for a specific protocol, use the name or number protocol as listed in <http://www.iana.org/assignments/protocol-numbers>.

- `ProtocolID`

IANA protocol number specified by the user.

- `QoS_AppliedInterface`

For rate-limited connections, the name of the interface where you applied rate limiting.

- `QoS_Policy`

The QoS policy that rate limited the connection.

- `QoS_Rule`

The QoS rule that rate limited the connection.

- `QoS_RuleID`

Internal ID number of the Quality of Service rule applied to the connection, if applicable.

- `RealmID`

The ID number of the realm. This field is the unique key for this record.

- `ReferencedHost`

If the protocol in the connection is HTTP or HTTPS, this field displays the host name that the respective protocol was using.

- `ResponderBytes`

The total number of bytes transmitted by the session initiator or received by the session responder.

- `ResponderBytesDropped`

The number of bytes dropped from the session initiator or session responder due to rate limiting.

- `ResponderContinent`

When a routable IP is detected, the continent associated with the IP address for the session initiator or responder.

- `ResponderContinentCode`

ISO-3166 Code for the continent of the destination host.

- `ResponderCountry`

When a routable IP is detected, the country associated with the IP address of the session initiator or responder. The system displays an icon of the country's flag, and the country's ISO 3166-1 alpha-3 country code. Hover your pointer over the flag icon to view the country's full name.

- `ResponderCountryCode`

ISO-3166 code for the country of the destination host.

- `ResponderCountryID`

Code for the country of the destination host.

- `ResponderIP`

In the Firepower Management Center web interface, these values constrain summaries and graphs.

The IP address (and host name, if DNS resolution is enabled) of the session initiator or responder.

See also [A Note About Initiator/Responder, Source/Destination, and Sender/Receiver Fields](#).

In the Firepower Management Center web interface, the host icon identifies the IP address that caused the connection to be blocked.

For plaintext, passthrough tunnels either blocked or fastpathed by the prefilter policy, initiator and responder IP addresses represent the tunnel endpoints—the routed interfaces of the network devices on either side of the tunnel.

- `ResponderPackets`

The total number of packets transmitted by the session initiator or received by the session responder.

- `ResponderPacketsDropped`

The number of packets dropped from the session initiator or session responder due to rate limiting.

- `ResponderPort`

Port used by the responding host.

- `SecurityGroupID`

ID number assigned to the user by ISE based on policy.

- `SensorID`

The identification number of the detecting managed device.

- `SI_Layer`

The IP layer that matched the IP block list.

- `SourceIP_DynamicAttribute`

Dynamic Attributes associated with the source IP address.

- `SourceSecurityGroupTagType`

How the Source Security Group Tag was assigned:

- 0 — Unknown
- 1 — Inline
- 2 — Session Directory
- 3 — Security Group Tag Exchange Protocol (SXP)

- `SSL_ActualAction`

In the Firepower Management Center web interface, this field is a search field only.

The system displays field values in the **SSL Status** field on search workflow pages.

The action the system applied to encrypted traffic in the SSL policy.

Action	Description
Block/Block with reset	Represents blocked encrypted connections.
Decrypt (Resign)	Represents an outgoing connection decrypted using a re-signed server certificate.
Decrypt (Replace Key)	Represents an outgoing connection decrypted using a self-signed server certificate with a substituted public key.
Decrypt (Known Key)	Represents an incoming connection decrypted using a known private key.
Default Action	Indicates the connection was handled by the default action.
Do not Decrypt	Represents a connection the system did not decrypt.

- `SSL_ActualActionID`

Code for the action performed on the connection based on the SSL Rule.

- `SSL_Cert`

The information stored on the public key certificate used to encrypt traffic, including:

- Subject/Issuer Common Name
- Subject/Issuer Organization
- Subject/Issuer Organization Unit
- Not Valid Before/After
- Serial Number
- Certificate Fingerprint
- Public Key Fingerprint

- `SSL_CertFingerprint`

SHA1 hash of the SSL Server certificate.

- `SSL_CipherSuite`

A macro value representing a cipher suite used to encrypt the connection. See <https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml> for cipher suite value designations.

- `SSL_ExpectedAction`

In the Firepower Management Center web interface, this field is a search field only.

The action the system expected to apply to encrypted traffic, given the SSL rule in effect.

Enter any of the values listed for **SSL Actual Action**.

- `SSL_FlowError`

The error name and hexadecimal code if an error occurred during the TLS/SSL session; `Success` if no error occurred.

- `SSL_FlowFlags`

The first ten debugging level flags for an encrypted connection. On a workflow page, to view all flags, click the ellipsis (...).

- `SSL_FlowMessages`

The keywords below indicate encrypted traffic is associated with the specified message type exchanged between client and server during the TLS/SSL handshake. See <http://tools.ietf.org/html/rfc5246> for more information.

- HELLO\_REQUEST
  - CLIENT\_ALERT
  - SERVER\_ALERT
  - CLIENT\_HELLO
  - SERVER\_HELLO
  - SERVER\_CERTIFICATE
  - SERVER\_KEY\_EXCHANGE
  - CERTIFICATE\_REQUEST
  - SERVER\_HELLO\_DONE
  - CLIENT\_CERTIFICATE
  - CLIENT\_KEY\_EXCHANGE
  - CERTIFICATE\_VERIFY
  - CLIENT\_CHANGE\_CIPHER\_SPEC
  - CLIENT\_FINISHED
  - SERVER\_CHANGE\_CIPHER\_SPEC
  - SERVER\_FINISHED
  - NEW\_SESSION\_TICKET
  - HANDSHAKE\_OTHER
  - APP\_DATA\_FROM\_CLIENT
  - APP\_DATA\_FROM\_SERVER
  - SERVER\_NAME\_MISMATCH
- The server certificate seen in the session has a Common Name or SAN values not corresponding to the destined domain name.
- CERTIFICATE\_CACHE\_HIT

A certificate matching the destined domain name was found in the cache.

- CERTIFICATE\_CACHE\_MISS

A certificate matching the destined domain name was not found in the cache.

- SSL\_FlowStatus

The reason the system failed to decrypt encrypted traffic:

- Unknown
- No Match
- Success
- Uncached Session
- Unknown Cipher Suite
- Unsupported Cipher Suite
- Unsupported SSL Version
- SSL Compression Used
- Session Undecryptable in Passive Mode
- Handshake Error
- Decryption Error
- Pending Server Name Category Lookup
- Pending Common Name Category Lookup
- Internal Error
- Incomplete Handshake
- Network Parameters Unavailable
- Invalid Server Certificate Handle
- Server Certificate Fingerprint Unavailable
- Cannot Cache Subject DN
- Cannot Cache Issuer DN
- Unknown SSL Version
- External Certificate List Unavailable
- External Certificate Fingerprint Unavailable
- Internal Certificate List Invalid
- Internal Certificate List Unavailable
- Internal Certificate Unavailable
- Internal Certificate Fingerprint Unavailable

- Server Certificate Validation Unavailable
- Server Certificate Validation Failure
- Invalid Action

- `SSL_Policy`

The SSL policy that handled the connection.

If TLS server identity discovery is enabled in the access control policy advanced settings, and there is no SSL policy associated with the access control policy, this field holds `none` for all SSL events.

- `SSL_PolicyUUID`

The UUID of the SSL Policy. This field is the unique key for this record.

- `SSL_Rule`

The SSL rule or default action that handled the connection, as well as the first Monitor rule matched by that connection. If the connection matched a Monitor rule, the field displays the name of the rule that handled the connection, followed by the Monitor rule name.

- `SSL_RuleID`

ID number of the SSL rule or default action that handled the connection.

- `SSL_Server`

Hostname of the server with which the client established an encrypted connection.

- `SSL_ServerCertStatus`

The SSL Server Certificate Status Number. This field is the unique key for this record.

- `SSL_SessionID`

The hexadecimal Session ID negotiated between the client and server during the TLS/SSL handshake.

- `SSL_TicketID`

A hexadecimal hash value of the session ticket information sent during the TLS/SSL handshake.

- `SSL_URL_Category`

URL categories for the URL visited in the encrypted connection.

This field exists ONLY as a syslog field; in the Firepower Management Center web interface, values in this field are included in the URL Category column.

- `SSL_Version`

The TLS/SSL protocol version used to encrypt the connection:

- Unknown
- SSLv2.0
- SSLv3.0
- TLSv1.0
- TLSv1.1

- `TLSv1.2`
- `TLSv1.3`
  
- `SSL_VersionID`

SSL Version ID number. This field is the unique key for this record.
- `TCP_Flags`

For connections generated from NetFlow data, the TCP flags detected in the connection.

When searching this field, enter a list of comma-separated TCP flags to view all connections that have *at least* one of those flags.
- `TunnelRule`

The tunnel rule, prefilter rule, or prefilter policy default action that handled the connection.
- `TunnelRuleID`

Internal identifier for the tunnel rule that triggered the event, if applicable.
- `URL`

The URL requested by the monitored host during the session and its associated category and reputation, if available.
- `URL_Category`

For an event to display URL category and reputation, you must include the applicable URL rules in an access control policy and configure the rule with URL category and URL reputation under the **URLs** tab.

URL category and reputation do not appear in an event if the connection is processed before it matches a URL rule.
- `URL_CategoryID`

ID number of the URL category. This field is the unique key for this record.
- `URL_Reputation`

For an event to display URL category and reputation, you must include the applicable URL rules in an access control policy and configure the rule with URL category and URL reputation under the **URLs** tab.

URL category and reputation do not appear in an event if the connection is processed before it matches a URL rule.
- `URL_ReputationLevel`

Reputation level assigned to the URL. This value is from 1 to 5, with 1 being untrusted and 5 being trusted.
- `UserAgent`

The user-agent string application information extracted from HTTP traffic detected in the connection.
- `UserID`

Internal identification number for the user who last logged into the host that generated the traffic.

- `UserName`

The username associated with the IP address of the host that initiated the connection, which may or may not be the source host of the exploit. This user value is typically known only for users on your network.

- `UserProtocol`

IANA protocol number specified by the user.

- `VLAN_ID`

The innermost VLAN ID associated with the packet that triggered the connection.

- `WebApplication`

The web application, which represents the content or requested URL for HTTP traffic detected in the connection.

If the web application does not match the URL for the event, the traffic is probably referred traffic, such as advertisement traffic. If the system detects referred traffic, it stores the referring application (if available) and lists that application as the web application.

If the system cannot identify the specific web application in HTTP traffic, this field displays `Web Browsing`.

- `WebApplicationHTTP`

If the system detects an application protocol of HTTP but cannot detect a specific web application, the system supplies a generic web browsing designation instead.

- `WebApplicationID`

The internal identification number of the detected web application, if applicable.

- `WebApplicationProductivityIndex`

Criteria that characterize the application to help you understand the application's function.

- `ZeroTrustApplication`

The Zero Trust Application detected in the event.

- `ZeroTrustApplicationGroup`

The Zero Trust Application Group to which the Zero Trust Application belongs.

- `ZeroTrustApplicationPolicy`

The Zero Trust Application policy triggered by the event.

## File Event

The File Event data block contains information on files that are sent over the network. This includes the connection information, whether the file is malware, and specific information to identify the file.

You can request fully-qualified file events from the eStreamer service by including a "FileEvent" section in the request JSON file. This section must specify the requested fields. The following fields may be requested for file events:

The following fields can be requested from File Events:



- `Application`

The client application accessing the malware file when AMP for Endpoints detection occurred. These applications are **not** tied to network discovery or application control.

- `ApplicationID`

ID number associated with the client application accessing the file.

- `ApplicationProductivityIndex`

The productivity associated with the application traffic detected in the connection.

- `ApplicationRiskIndex`

The risk associated with detected applications in the traffic that triggered the intrusion event: Very High, High, Medium, Low, and Very Low. Each type of application detected in a connection has an associated risk; this field displays the highest risk of those.

- `ArchiveDepth`

The level (if any) at which the file was nested in an archive file.

- `ArchiveFileName`

The name of the archive file (if any) which contained the malware file.

- `ArchiveFileStatus`

The status of an archive being inspected. Can have the following values:

- Pending — Archive is being inspected
- Extracted — Successfully inspected without any problems
- Failed — Failed to inspect, insufficient system resources
- Depth Exceeded — Successful, but archive exceeded the nested inspection depth
- Encrypted — Partially successful, archive was or contains an archive that is encrypted
- Not Inspectable — Partially successful, file is possibly malformed or corrupt

- `ArchiveFileStatusID`

ID number associated with the Archive File Status.

- `ArchiveSHA256`

The SHA-256 hash value of the archive file (if any) which contains the malware file.

- `AuthenticationSource`

Type of authentication used by the user. Values may be:

- 0 - no authorization required
- 1 - passive authentication, AD agent, or ISE session
- 2 - captive portal successful authentication
- 3 - captive portal guest authentication
- 4 - captive portal failed authentication

- `ClientApplication`

The client application, if available, which represents software running on the monitored host detected in the traffic that triggered the intrusion event.

- `ClientApplicationID`

The ID number associated with the client application.

- `ClientApplicationProductivityIndex`

The productivity associated with the client application traffic detected in the connection.

- `ClientApplicationRiskIndex`

Risk value of the client application. Values may be:

- 1 - Very Low
- 2 - Low
- 3 - Medium
- 4 - High
- 5 - Very High

- `ConnectionID`

A unique identifier for the connection with the server.

- `Context`

The metadata identifying the virtual firewall group through which the traffic passed. The system only populates this field for ASA FirePOWER in multiple context mode.

- `ContextUUID`

UUID associated with the virtual firewall security context.

- `Device`

For file events and for malware events generated by Firepower devices, the name of the device that detected the file.

For malware events generated by AMP for Endpoints and for retrospective malware events generated by the AMP cloud, the name of the FMC.

- `DeviceIP`

IP Address of the device which detected the event.

- `DeviceSerialNumber`

Serial number of the device which detected the event.

- `DeviceUUID`

The unique identifier of the Firepower device that generated an event.

For malware events generated by AMP for Endpoints and for retrospective malware events generated by the AMP cloud, the name of the FMC.

- `Domain`

For file events and for malware events generated by Firepower devices, the domain of the device that detected the file. For malware events generated by AMP for Endpoints and for retrospective malware events generated by the AMP cloud, the domain associated with the AMP cloud connection that reported the event.

This field is only present if you have ever configured the FMC for multitenancy.

- `EgressVRF`

In networks using virtual routing, the name of the virtual router through which traffic exited the network.

- `EventDescription`

The additional event information associated with the event type.

- `EventID`

ID number assigned to the event.

- `EventPriority`

Priority assigned to the event. This is an integer value from 0 to 5.

- `EventSecond`

UNIX timestamp (seconds since 01/01/1970) of the event's detection.

- `EventSubtype`

The AMP for Endpoints action that led to malware detection, for example, Create, Execute, Move, or Scan.

- `EventType`

The sub-type of malware event.

- `FileDirection`

Whether the file was downloaded or uploaded during the connection. Possible values are:

- Download — the file was transferred from the DstIP to the SrcIP.
- Upload — the file was transferred from the SrcIP to the DstIP.

- `FileName`

The name of the file.

- `FilePolicy`

The file policy that detected the file.

- `FileStorageStatus`

The storage status of the file associated with the event:

**Stored**

Returns all events where the associated file is currently stored.

### Stored in connection

Returns all events where the system captured and stored the associated file, regardless of whether the associated file is currently stored.

### Failed

Returns all events where the system failed to store the associated file.

- `FileType`  
The type of file, for example, HTML or MSEXE.
- `FirstPacketSecond`  
The time at which the file download or upload flow started.
- `Hostname`  
Domain name of the detected HTTP request.
- `HTTP_Response`  
The HTTP status code sent in response to a client's HTTP request when a file is transferred.
- `IngressVRF`  
In networks using virtual routing, the name of the virtual router through which traffic entered the network.
- `InitiatorContinent`  
When a routable IP is detected, the continent associated with the IP address for the session initiator or responder.
- `InitiatorContinentCode`  
ISO-3166 code for the continent of the source host.
- `InitiatorCountry`  
When a routable IP is detected, the country associated with the IP address of the session initiator or responder. The system displays an icon of the country's flag, and the country's ISO 3166-1 alpha-3 country code. Hover your pointer over the flag icon to view the country's full name.
- `InitiatorCountryCode`  
Code for the country of the source host.
- `InitiatorCountryID`  
ISO-3166 value for the country of the initiating host.
- `InitiatorIP`  
The IP address of the host that initiated the connection. This may be the IP address of the sender or the recipient of the file, depending on the value in the `FileDirection` field:
- `InitiatorPort`  
The port or ICMP type used by the session initiator.
- `InstanceID`  
The Snort instance that processed the connection event. This field has no significance on its own.

- `ManagerName`  
Name of the Secure Firewall Management Center which detected the event.
- `ManagerUUID`  
UUID of the Secure Firewall Management Center which detected the event.
- `MatchedRule`  
The specific rule which triggered the event.
- `MitreAttackGroups`  
A count of techniques that you can click to bring up a modal, which shows the full list of MITRE tactics and techniques within that hierarchy.
- `NetmapID`  
The first bit of this field is a flag indicating whether the header is an extended header containing an archive timestamp. The remaining 15 bits are an optional field containing the Netmap ID for the domain on which the event was detected. If this field is not used, it is left empty. Netmap IDs map to domains as provided in metadata.
- `Protocol`  
The protocol used for the connection, for example TCP or UDP.
- `ProtocolID`  
IANA protocol number specified by the user.
- `RealmID`  
The ID number of the realm.
- `RealmName`  
The realm name.
- `ReceivingContinent`  
The continent of the host receiving the file.
- `ReceivingCountry`  
The country of the host receiving the file.
- `ReceivingCountryCode`  
ISO-3166 code for the country of the destination host.
- `ReceivingCountryID`  
Code for the country of the destination host.
- `ReceivingIP`  
In the FMC web interface, for file events and for malware events generated by Firepower devices, the IP address of the host receiving the file. See also [A Note About Initiator/Responder, Source/Destination, and Sender/Receiver Fields](#).  
  
For malware events generated by AMP for Endpoints, the IP address of the endpoint whose connector reported the event.

For syslog equivalents (events generated by Firepower devices only), see **DstIP** and **SrcIP**.

- `ReceivingPort`

In the FMC web interface, the destination port used by the traffic where the file was detected.

For syslog equivalents, see **DstIP** and **SrcIP** and **DstPort** and **SrcPort**.

- `ResponderContinent`

When a routable IP is detected, the continent associated with the IP address for the session initiator or responder.

- `ResponderContinentCode`

ISO-3166 Code for the continent of the destination host.

- `ResponderCountry`

When a routable IP is detected, the country associated with the IP address of the session initiator or responder. The system displays an icon of the country's flag, and the country's ISO 3166-1 alpha-3 country code. Hover your pointer over the flag icon to view the country's full name.

- `ResponderCountryCode`

ISO-3166 code for the country of the destination host.

- `ResponderCountryID`

Code for the country of the destination host.

- `ResponderIP`

The IP address (and host name, if DNS resolution is enabled) of the session initiator or responder.

- `ResponderPort`

Port used by the responding host.

- `SendingContinent`

The continent of the host sending the file.

- `SendingCountry`

The country of the host sending the file.

- `SendingIP`

In the FMC web interface, the IP address of the host sending the file. See also [A Note About Initiator/Responder, Source/Destination, and Sender/Receiver Fields](#).

For syslog equivalents, see **DstIP** and **SrcIP**.

- `SendingPort`

In the FMC web interface, the source port used by the traffic where the file was detected.

For syslog equivalents, see **DstIP** and **SrcIP** and **DstPort** and **SrcPort**.

- `SensorID`

The identification number of the detecting managed device.

- `SHA_Disposition`

The file's disposition:

**Malware**

Indicates that the AMP cloud categorized the file as malware, local malware analysis identified malware, or the file's threat score exceeded the malware threshold defined in the file policy.

**Clean**

Indicates that the AMP cloud categorized the file as clean, or that a user added the file to the clean list. Clean files appear in the malware table only if they were changed to clean.

**Unknown**

Indicates that the system queried the AMP cloud, but the file has not been assigned a disposition; in other words, the AMP cloud has not categorized the file.

**Custom Detection**

Indicates that a user added the file to the custom detection list.

**Unavailable**

Indicates that the system could not query the AMP cloud. You may see a small percentage of events with this disposition; this is expected behavior.

File dispositions appear only for files for which the system queried the AMP cloud.

- `SperoDisposition`

Indicates whether the SPERO signature was used in file analysis. Possible values:

- Spero detection performed on file
- Spero detection not performed on file

- `SSL_ActualAction`

The action the system applied to encrypted traffic:

**Block or Block with reset**

Represents blocked encrypted connections.

**Decrypt (Resign)**

Represents an outgoing connection decrypted using a re-signed server certificate.

**Decrypt (Replace Key)**

Represents an outgoing connection decrypted using a self-signed server certificate with a substituted public key.

**Decrypt (Known Key)**

Represents an incoming connection decrypted using a known private key.

**Default Action**

Indicates the connection was handled by the default action.

**Do not Decrypt**

Represents a connection the system did not decrypt.

- `SSL_Cert`

The information stored on the public key certificate used to encrypt traffic, including:

- Subject/Issuer Common Name
- Subject/Issuer Organization
- Subject/Issuer Organization Unit
- Not Valid Before/After
- Serial Number
- Certificate Fingerprint
- Public Key Fingerprint

- `SSL_CertFingerprint`

The certificate fingerprint of the TLS/SSL server.

- `SSL_FlowStatus`

The reason the system failed to decrypt encrypted traffic:

- Unknown
- No Match
- Success
- Uncached Session
- Unknown Cipher Suite
- Unsupported Cipher Suite
- Unsupported SSL Version
- SSL Compression Used
- Session Undecryptable in Passive Mode
- Handshake Error
- Decryption Error
- Pending Server Name Category Lookup
- Pending Common Name Category Lookup
- Internal Error
- Network Parameters Unavailable
- Invalid Server Certificate Handle
- Server Certificate Fingerprint Unavailable
- Cannot Cache Subject DN
- Cannot Cache Issuer DN



- Unknown SSL Version
  - External Certificate List Unavailable
  - External Certificate Fingerprint Unavailable
  - Internal Certificate List Invalid
  - Internal Certificate List Unavailable
  - Internal Certificate Unavailable
  - Internal Certificate Fingerprint Unavailable
  - Server Certificate Validation Unavailable
  - Server Certificate Validation Failure
  - Invalid Action
- 
- `ThreatName`

The name of the detected malware.
  - `ThreatScore`

The threat score most recently associated with this file. This is a value from 0 to 100 based on the potentially malicious behaviors observed during dynamic analysis.
  - `URI`

The URI of the connection associated with the file transaction, for example, the URL from which a user downloaded the file.
  - `UserName`

The username associated with the IP address that initiated the connection. If this IP address is external to your network, the associated username is typically unknown.
  - `UserNameNoRealm`

For file events and for malware events generated by Firepower devices, this field displays the username that was determined by an identity policy or authoritative logins. In absence of an identity policy, it displays No Authentication Required.
  - `WebApplication`

The application that represents the content or requested URL for HTTP traffic detected in the connection.

## Intrusion Event

Intrusion events are generated when the system detects a possible intrusion. An intrusion event is a record of the date, time, the type of exploit, and contextual information about the source of the attack and its target.

You can request fully-qualified intrusion events from the eStreamer service by including an "IntrusionEvent" section in the request JSON file. This section must specify the requested fields. The following fields may be requested for intrusion events:

- `Application`

The application protocol, if available, which represents communications between hosts detected in the traffic that triggered the intrusion event.

- `ApplicationID`

ID number that maps to the application using the file transfer.

- `ApplicationProductivityIndex`

Indicates criticality of host to business.” (Assuming this is the same as the “business criticality” field.

- `ApplicationRiskIndex`

The risk associated with detected applications in the traffic that triggered the intrusion event: Very High, High, Medium, Low, and Very Low. Each type of application detected in a connection has an associated risk; this field displays the highest risk of those.

- `AuthenticationSource`

- `Classification`

The classification where the rule that generated the event belongs.

See a list of possible classification values in [Intrusion Event Details](#).

When searching this field, enter the classification number, or all or part of the classification name or description for the rule that generated the events you want to view. You can also enter a comma-separated list of numbers, names, or descriptions. Finally, if you add a custom classification, you can also search using all or part of its name or description.

- `ClientApplication`

The client application, if available, which represents software running on the monitored host detected in the traffic that triggered the intrusion event.

- `ClientApplicationID`

The internal identification number of the detected client application, if applicable.

- `ClientApplicationProductivityIndex`

Business relevance of the application. Values may be:

- 1 - Very Low
- 2 - Low
- 3 - Medium
- 4 - High
- 5 - Very High

- `ClientApplicationRiskIndex`

Risk value of the client application. Values may be:

- 1 - Very Low
- 2 - Low

- 3 - Medium
- 4 - High
- 5 - Very High

- `ConnectionID`

Numerical ID of the Snort instance on the managed device that generated the connection event.

- `Context`

The metadata identifying the virtual firewall group through which the traffic passed. The system only populates this field for ASA FirePOWER in multiple context mode.

- `CVE_ID`

This field is a search field only.

Search by the identification number associated with the vulnerability in MITRE's Common Vulnerabilities and Exposures (CVE) database (<https://cve.mitre.org/>).

- `Device`

The managed device where the access control policy was deployed.

- `DeviceIP`

IP address of the sensor which detected the event.

- `DeviceSerialNumber`

Serial number of the sensor which detected the event.

- `DeviceUUID`

The unique identifier of the Firepower device that generated an event.

The following fields collectively uniquely identify the connection event associated with a particular intrusion event: DeviceUUID, First Packet Time, Connection Instance ID, and Connection Counter.

- `Domain`

The domain of the device that detected the intrusion. This field is only present if you have ever configured the FMC for multitenancy.

- `EgressInterface`

The egress interface of the packet that triggered the event. This interface column is not populated for a passive interface.

- `EgressInterfaceUUID`

An interface ID number that acts as a unique identifier for the egress interface.

- `EgressVRF`

In networks using virtual routing, the names of the virtual routers through which traffic entered and exited the network.

- `EgressZone`

The egress security zone of the packet that triggered the event. This security zone field is not populated in a passive deployment.

- `EgressZoneUUID`

A zone ID that acts as the unique identifier for the egress security zone associated with correlation event.

- `EventID`

Unique ID number of the event.

- `EventMicrosecond`

Microsecond (one millionth of a second) increment of the timestamp of the event's detection.

- `EventPriority`

Whether or not the connection event is a high priority event. High priority events are connection events that are associated with an intrusion, Security Intelligence, file, or malware event. All other events are Low priority.

- `EventSecond`

UNIX timestamp (seconds since 01/01/1970) of the event's detection.

- `EventType`

The sub-type of malware event.

- `FirewallPolicy`

The access control policy containing the access control rule which triggered the event.

- `FirewallPolicyUUID`

A policy ID number that acts as a unique identifier for the access control policy.

- `FirewallRule`

Access control rule which triggered the event.

- `FirewallRuleID`

A rule ID number that acts as a unique identifier for the access control rule.

- `FirstPacketSecond`

UNIX timestamp of the date and time the first packet was exchanged in the session.

- `GeneratorID`

Generator ID; the ID of the component that generated the event.

See also information about the following intrusion event fields: Generator, Message, and Snort ID.

- `Hostname`

The host name found in the HTTP connection.

- `HTTP_Hostname`

The host name, if present, that was extracted from the HTTP request Host header. Note that request packets do not always include the host name.

To associate host names with intrusion events for HTTP client traffic, you must enable the HTTP Inspect preprocessor **Log Hostname** option.

In table views, this column displays the first fifty characters of the extracted host name. You can hover your pointer over the displayed portion of an abbreviated host name to display the complete name, up to 256 bytes. You can also display the complete host name, up to 256 bytes, in the packet view.

- `HTTP_Response`

Response code of the HTTP Request.

- `HTTP_URI`

The raw URI, if present, associated with the HTTP request packet that triggered the intrusion event. Note that request packets do not always include a URI.

To associate URIs with intrusion events for HTTP traffic, you must enable the HTTP Inspect preprocessor **Log URI** option.

To see the associated HTTP URI in intrusion events triggered by HTTP responses, you should configure HTTP server ports in the **Perform Stream Reassembly on Both Ports** option; note, however, that this increases resource demands for traffic reassembly.

This column displays the first fifty characters of the extracted URI. You can hover your pointer over the displayed portion of an abbreviated URI to display the complete URI, up to 2048 bytes. You can also display the complete URI, up to 2048 bytes, in the packet view.

- `ICMP_Code`

The port or ICMP code used by the session responder.

- `ICMP_Type`

In the Firepower Management Center web interface, these values constrain summaries and graphs.

The port or ICMP type used by the session initiator.

- `Impact`

The impact level in this field indicates the correlation between intrusion data, network discovery data, and vulnerability information.

When searching this field, do not specify impact icon colors or partial strings. For example, do not use **blue**, **level 1**, or **0**. Valid case-insensitive values are:

- Impact 0, Impact Level 0
- Impact 1, Impact Level 1
- Impact 2, Impact Level 2
- Impact 3, Impact Level 3
- Impact 4, Impact Level 4

Because no operating system information is available for hosts added to the network map from NetFlow data, the system cannot assign Vulnerable (impact level 1: red) impact levels for intrusion events involving those hosts. In such cases, use the host input feature to manually set the operating system identity for the hosts.

- `ImpactFlag`

Impact flag value of the event. The low-order eight bits indicate the impact level. Values are:

- 0x01 (bit 0) — Source or destination host is in a network monitored by the system.
- 0x02 (bit 1) — Source or destination host exists in the network map.
- 0x04 (bit 2) — Source or destination host is running a server on the port in the event (if TCP or UDP) or uses the IP protocol.
- 0x08 (bit 3) — There is a vulnerability mapped to the operating system of the source or destination host in the event.
- 0x10 (bit 4) — There is a vulnerability mapped to the server detected in the event.
- 0x20 (bit 5) — The event caused the managed device to drop the session (used only when the device is running in inline, switched, or routed deployment). Corresponds to blocked status in the Secure Firewall System web interface.
- 0x40 (bit 6) — The rule that generated this event contains rule metadata setting the impact flag to red. The source or destination host is potentially compromised by a virus, trojan, or other piece of malicious software.
- 0x80 (bit 7) — There is a vulnerability mapped to the client detected in the event. (version 5.0+ only)

The following impact level values map to specific priorities on the Management Center. An X indicates the value can be 0 or 1:

- gray (0, unknown): 00X00000
- red (1, vulnerable): XXXX1XXX, XXX1XXXX, X1XXXXXX, 1XXXXXXX (version 5.0+ only)
- orange (2, potentially vulnerable): 00X0011X
- yellow (3, currently not vulnerable): 00X0001X
- blue (4, unknown target): 00X00001

- `IngressInterface`

The ingress interface of the packet that triggered the event. Only this interface column is populated for a passive interface.

- `IngressInterfaceUUID`

An interface ID number that acts as a unique identifier for the ingress interface.

- `IngressVRF`

In networks using virtual routing, the names of the virtual routers through which traffic entered and exited the network.

- `IngressZone`


The ingress security zone or tunnel zone of the packet that triggered the event. Only this security zone field is populated in a passive deployment.



- `IngressZoneUUID`

Ingress security zone in the event that triggered the policy violation.

- `InitiatorContinent`  
The continent of the sending host involved in the intrusion event.
- `InitiatorContinentCode`  
ISO-3166 code for the continent of the source host.
- `InitiatorCountry`  
The country of the sending host involved in the intrusion event.
- `InitiatorCountryCode`  
Code for the country of the source host.
- `InitiatorCountryID`  
ISO-3166 value for the country of the initiating host.
- `InitiatorHostCriticality`  
User-defined criticality value for the source host:
  - 0 — None
  - 1 — Low
  - 2 — Medium
  - 3 — High
- `InitiatorIP`  
The IP address used by the sending host involved in the intrusion event.  
See also [A Note About Initiator/Responder, Source/Destination, and Sender/Receiver Fields](#).
- `InitiatorOS_Name`  
Operating system of the source host.
- `InitiatorOS_Version`  
Version number of the source host's operating system.
- `InitiatorPort`  
The port number on the sending host. For ICMP traffic, where there is no port number, this field displays the ICMP type.
- `InlineResult`  
In workflow and table views, this field displays one of the following:

**Table 2: Inline Result Field Contents in Workflow and Table Views**

This Icon	Indicates
	The system dropped the packet that triggered the rule.

This Icon	Indicates
	IPS would have dropped the packet if you enabled the <b>Drop when Inline</b> intrusion policy option (in an inline deployment), or if a Drop and Generate rule generated the event while the system was pruning.
	IPS may have transmitted or delivered the packet to the destination, but the connection that contained this packet is now blocked.
No icon (blank)	The triggered rule was not set to Drop and Generate Events

The following table lists the possible reasons for the inline results — Would have dropped and Partially dropped.

Inline Result	Reason	Detailed Reason
Would Have Dropped	Interface in Passive or Tap mode	You have configured the interfaces in inline tap or passive mode.
	Intrusion Policy in "Detection" Inspection Mode	You have set the inspection mode in the intrusion policy to Detection.
	Connection Timed Out	The Snort inspection engine has suspended the inspection as the TCP/IP connection timed out.
Partially Dropped	Connection Closed (0x01)	While creating a new flow, if the allocated flows are more than the allowed number of flows, the Snort inspection engine prunes the least recently used flows.
	Connection Closed (0x02)	When reloading the Snort inspection engine causes a memory adjustment, the engine prunes the least recently used flows.
	Connection Closed (0x04)	When the Snort inspection engine is gracefully shutting down, the engine purges all the active flows.

In a passive deployment, the system does not drop packets, including when an inline interface is in tap mode, regardless of the rule state or the inline drop behavior of the intrusion policy.

When searching this field, enter either of the following:

- **dropped** to specify whether the packet is dropped in an inline deployment.
- **would have dropped** to specify whether the packet would have dropped if the intrusion policy had been set to drop packets in an inline deployment.



- **partially dropped** to specify whether the packet is transmitted or delivered to the destination, but the connection that contained this packet is now blocked.

- `InlineResultID`

Value which maps to the Inline Result.

- `InlineResultReason`

Value indicating the inline result reason.

- 1— Interface in Passive or Tap mode
- 2— Intrusion Policy in “Detection” inspection mode
- 3— Network Analysis Policy in “Detection” inspection mode
- 4— Connection timed out
- 5— Connection Closed (internal use)
- 6— Connection Closed (internal use)
- 7— Connection Closed (internal use)

- `InlineResultReasonID`

Value which maps to the Inline Result.

- `InstanceID`

Numerical ID of the Snort instance on the managed device that generated the event.

- `IntrusionPolicy`

The intrusion policy where the intrusion, preprocessor, or decoder rule that generated the event was enabled. You can choose an intrusion policy as the default action for an access control policy, or you can associate an intrusion policy with an access control rule.

- `IntrusionPolicyRevUUID`

The unique identifier for the intrusion policy associated with the connection event. This field is the unique key for this record.

- `IntrusionPolicyUUID`

The unique identifier for the intrusion policy associated with the connection event. This field is the unique key for this record.

- `IntrusionRuleMessage`

Rule message that triggered the event.

- `ManagerName`

Name of the Secure Firewall Management Center which detected the event.

- `ManagerUUID`

UUID of the Secure Firewall Management Center which detected the event.

- `MatchedRule`

The specific rule which triggered the event.

- `MPLS_Label`

The Multiprotocol Label Switching label associated with the packet that triggered the intrusion event.

- `NAP_Policy`

The network analysis policy, if any, associated with the generation of the event.

This field displays the first fifty characters of the extracted URI. You can hover your pointer over the displayed portion of an abbreviated URI to display the complete URI, up to 2048 bytes. You can also display the complete URI, up to 2048 bytes, in the packet view.

- `NAP_PolicyUUID`

The UUID of the Network Analysis Policy that created the intrusion event.

- `NetmapID`

The first bit of this field is a flag indicating whether the header is an extended header containing an archive timestamp. The remaining 15 bits are an optional field containing the Netmap ID for the domain on which the event was detected. If this field is not used, it is left empty. Netmap IDs map to domains as provided in metadata.

- `OriginalInitiatorIP`

Contains the IP address of the original initiator of the connection.

- `Priority`

The event priority as determined by the Talos Intelligence Group. The priority corresponds to either the value of the `priority` keyword or the value for the `classtype` keyword. For other intrusion events, the priority is determined by the decoder or preprocessor. Valid values are high, medium, and low.

- `PriorityID`

Identification number of the priority associated with the event.

- `Protocol`

In the Firepower Management Center web interface, this field is a search field only.

The name or number of the transport protocol used in the connection as listed in <http://www.iana.org/assignments/protocol-numbers>. This is the protocol associated with the source and destination port/ICMP column.

- `ProtocolID`

Identification number of the priority associated with the event.

- `RealmID`

The ID number of the realm. This field is the unique key for this record.

- `RealmName`

Name of the realm

- `ResponderContinent`

The continent of the receiving host involved in the intrusion event.

- `ResponderContinentCode`  
ISO-3166 Code for the continent of the destination host.
- `ResponderCountry`  
The country of the receiving host involved in the intrusion event.
- `ResponderCountryCode`  
ISO-3166 code for the country of the destination host.
- `ResponderCountryID`  
Code for the country of the destination host.
- `ResponderHostCriticality`  
User-defined criticality value for the destination host:
  - 0 — None
  - 1 — Low
  - 2 — Medium
  - 3 — High
- `ResponderIP`  
The IP address used by the receiving host involved in the intrusion event.  
See also [A Note About Initiator/Responder, Source/Destination, and Sender/Receiver Fields](#).
- `ResponderOS_Name`  
Operating system of the destination host.
- `ResponderOS_Version`  
Version number of destination host's operating system.
- `ResponderPort`  
Port used by the responding host.
- `ReviewedBy`  
The name of the user who reviewed the event. When searching this field, you can enter **unreviewed** to search for events that have not been reviewed.
- `SensorID`  
The identification number of the detecting managed device.
- `SignatureID`  
The signature ID (also known as the Snort ID) of the rule that generated the event.  
See also information about the following intrusion event fields: Generator, GID, Message, Revision, and Snort ID.
- `SMTP_Attachments`

Contains the MIME attachment file name that was extracted from the MIME Content-Disposition header. For this field to be populated you must enable the SMTP preprocessor Log MIME Attachment Names option. Multiple attachment file names are supported.

- SMTP\_From

Contains the address of the email sender that was extracted from the SMTP MAIL FROM command. For this field to be populated you must enable the SMTP preprocessor Log From Address option. Multiple sender addresses are supported.

- SMTP-Headers

Contains the data that was extracted from the email header. To associate email headers with intrusion events for SMTP traffic you must enable the SMTP preprocessor Log Headers option.

- SMTP\_To

Contains the address of the email recipient that was extracted from the SMTP RCPT TO command. For this field to be populated you must enable the SMTP preprocessor Log To Addresses option. Multiple recipient addresses are supported.

- SnortVersionID

Snort version number.

- SSL\_ActualAction

In the Firepower Management Center web interface, this field is a search field only.

The action the system applied to encrypted traffic:

**Block/Block with reset**

Represents blocked encrypted connections.

**Decrypt (Resign)**

Represents an outgoing connection decrypted using a re-signed server certificate.

**Decrypt (Replace Key)**

Represents an outgoing connection decrypted using a self-signed server certificate with a substituted public key.

**Decrypt (Known Key)**

Represents an incoming connection decrypted using a known private key.

**Default Action**

Indicates the connection was handled by the default action.

**Do not Decrypt**

Represents a connection the system did not decrypt.

Field values are displayed in the **SSL Status** field on the search workflow pages.

- SSL\_ActualActionID

Code for the action performed on the connection based on the SSL Rule.

- SSL\_Cert

The information stored on the public key certificate used to encrypt traffic, including:

- Subject/Issuer Common Name
  - Subject/Issuer Organization
  - Subject/Issuer Organization Unit
  - Not Valid Before/After
  - Serial Number
  - Certificate Fingerprint
  - Public Key Fingerprint
- `SSL_CertFingerprint`

SHA1 hash of the SSL Server certificate.

- `SSL_FlowStatus`

The reason the system failed to decrypt encrypted traffic:

- Unknown
- No Match
- Success
- Uncached Session
- Unknown Cipher Suite
- Unsupported Cipher Suite
- Unsupported SSL Version
- SSL Compression Used
- Session Undecryptable in Passive Mode
- Handshake Error
- Decryption Error
- Pending Server Name Category Lookup
- Pending Common Name Category Lookup
- Internal Error
- Incomplete Handshake
- Network Parameters Unavailable
- Invalid Server Certificate Handle
- Server Certificate Fingerprint Unavailable
- Cannot Cache Subject DN
- Cannot Cache Issuer DN

- Unknown SSL Version
- External Certificate List Unavailable
- External Certificate Fingerprint Unavailable
- Internal Certificate List Invalid
- Internal Certificate List Unavailable
- Internal Certificate Unavailable
- Internal Certificate Fingerprint Unavailable
- Server Certificate Validation Unavailable
- Server Certificate Validation Failure
- Invalid Action

- `SSL_FlowStatusID`

Code for the status of the SSL Flow.

- `UserID`

The internal identification number for the user, if applicable.

- `UserName`

The username associated with the IP address of the host that initiated the connection, which may or may not be the source host of the exploit. This user value is typically known only for users on your network.

- `UserProtocol`

IANA protocol number specified by the user.

- `VLAN_ID`

The innermost VLAN ID associated with the packet that triggered the intrusion event.

- `WebApplication`

The web application, which represents the content or requested URL for HTTP traffic detected in the traffic that triggered the intrusion event.

If the system detects an application protocol of HTTP but cannot detect a specific web application, the system supplies a generic web browsing designation instead.

- `WebApplicationHTTP`

If the system detects an application protocol of HTTP but cannot detect a specific web application, the system supplies a generic web browsing designation instead.

- `WebApplicationID`

The internal identification number for the web application, if applicable.

- `WebApplicationProductivityIndex`

Criteria that characterize the application to help you understand the application's function.

# Intrusion Packets

Intrusion Packets contain captured network traffic which triggered an Intrusion Event. The Intrusion Packets can be correlated with the corresponding Intrusion Event.

You can request fully-qualified Intrusion Packets from the eStreamer service by including an "IntrusionPacket" section in the request JSON file. This section must specify the requested fields. The following fields may be requested for intrusion packets:

- `Device`

The managed device where the access control policy was deployed.

The managed device that detected the connection or, for connections generated from NetFlow data, the managed device that processed the data.

- `DeviceIP`

IP address of the device which detected the event.

- `DeviceSerialNumber`

Serial number of the device which detected the event.

- `DeviceUUID`

The unique identifier of the Firepower device that generated an event.

The following fields collectively uniquely identify the connection event associated with a particular intrusion event: DeviceUUID, First Packet Time, Connection Instance ID, and Connection Counter.

- `Domain`

The domain of the device that detected the intrusion. This field is only present if you have ever configured the FMC for multitenancy.

- `EventID`

Unique ID number of the intrusion event.

- `EventSecond`

The second (from 01/01/1970) that the event occurred.

- `EventType`

The description of the event that triggered the correlation rule.

- `Hostname`

Domain name of the detected HTTP request.

- `ManagerName`

Name of the Secure Firewall Management Center which detected the event.

- `ManagerUUID`

UUID of the Secure Firewall Management Center which detected the event.

- `NetmapID`

The first bit of this field is a flag indicating whether the header is an extended header containing an archive timestamp. The remaining 15 bits are an optional field containing the Netmap ID for the domain on which the event was detected. If this field is not used, it is left empty. Netmap IDs map to domains as provided in metadata.

- `PacketData`

Actual captured packet data (header and payload).

- `PacketLength`

Number of bytes included in the packet data.

- `PacketMicrosecond`

Microsecond (one millionth of a second) increment that the packet was captured.

- `PacketSecond`

UNIX timestamp of the date and time the first packet was exchanged in the session.

- `SensorID`

The identification number of the detecting managed device.